Computer Network and Applications(19T2) - Assignment 1
Yash Umeshkumar Tamakuwala
z5248584

Python Version used - python3

Implementation of LSR

A router reads from the configuration file (configA.txt for example) provided to it. This file has information of the name of the router, the port it has to listen on, number of neighbours, name of neighbours, cost to reach them and the ports they would listen on. This information is essentially its Link State. Every router sends 2 types of messages to its neighbours:-

a. LSA- A router will advertise its link state to its neighbours. Since the message could be sent from any port (provided at random by OS), it will have itself as the ‘sender’ and ‘time’ at which it sends the message along with the link state to its neighbours. This will be done every second.

b. Forward LSA- A router will continuously listen for and receive LSA(of other routers) which it needs to forward to its neighbours. A router maintains information about the last time it received LSA of any router. Comparing this time with the timestamp in the message it will see if the message received is a new one or a stale one. If it is new, then it will put itself as the ‘FORWARDER’ and forward it to its neighbours else don’t process further.

After sometime, it will have an idea about the global topology and implement Dijkstra’s algorithm to find the least cost path to other routers. This happens every 30 seconds.

2. Data Structures used, LSA packet format, dealing with failed nodes, limiting excessive link-state broadcasts.

The network topology is maintained as a dictionary of dictionary (nested dictionaries) like - {'F': {'A': 2.2, 'D': 0.7, 'E': 6.2}, 'B': {'A': 6.5, 'C': 1.1, 'D': 4.2, 'E': 3.2}, 'E': {'B': 3.2, 'D': 2.9, 'F': 6.2}, 'A': {'B': 6.5, 'F': 2.2}, 'D': {'F': 0.7, 'B': 4.2, 'C': 1.6, 'E': 2.9}, 'C': {'B': 1.1, 'D': 1.6}}

It follows as router as key and its neighbours and cost to reach in a dictionary as the value. Link state packet is of the form -

```
{‘neighbours': {'B': 6.5, 'F': 2.2}, 'sender': 'A', 'time': 1564900562.152833}
```

This dictionary will also have a ‘FORWARDER’ field if a router is not the creator of the message but has to forward it. The receiver would receive a tuple of LSA message and sender’s - IP and port.

The program periodically checks for dead routers. They are those routers whose LSA it hasn’t received in some time(3-4s). This can be checked by seeing the time difference between the current time and the time it maintains. If the maintained time is older by 3-4s (for neighbours) or 13-14s (for distant routers), it means that the router is dead. This router then shall be removed from its neighbours and the global topology. When a router is dead, all its occurrences will be removed the link dictionary a node maintains and from list of neighbours. However its (listening)port information will be retained as a router can send through any port and that couldn’t be used to identify it.

Link state broadcasts are being limited as:-

a. If the timestamp in the message is older than the timestamp the router maintains for all the routers of the topology then it will consider this message as old and won’t process it further.

b. If the message is found to be new then the router will try to forward it. However, it will only forward to those neighbours that are neither the original sender nor the FORWARDER(both fields maintained and present in the message).

3. Design trade-offs considered, Unique Implementation, Scope of Improvement

Trade-offs:-

a. String processing is involved but dict() is a very convenient and flexible data structure.

b. Every message is decoded and needs to be converted as bytes->string->dictionary.

Despite the processing, the implementation remains special in the fact that :-

a. LSA message is always constructed from the current neighbours. Change in topology is almost instantly reflected in the next Link State Advertisement.

b. The main content of the message that is the neighbours information is sent in such a way that it can be directly used as is to form the global topology.

c. Time check for dead neighbours has been taken as (3-4s) and for distant neighbours as (13s) which takes into consideration that there will be only 10 nodes and situated in worst possible way - straight line.

d. Updating 'lastReceived' time for a forwarding router. A forwarding router is one that forwards an LSA it received. The 'lastReceived' time objective is to see when a router last sent its LSA but basically to see if it is alive or dead. However, this can also be used for forwarding router as a router can only transmit if it is alive.

e. If the received message has no FORWARDER it means that the sender is its neighbour. This information is useful especially when a dead router comes back alive.

Possible Improvements :-

a. Number of LS broadcasts could be further limited by not broadcasting to the common neighbours of the router and sender and router and FORWARDER. But this assumes that messages from the sender and FORWARDER have correctly reached and processed by their respective neighbours. We cant know that. This implies redundant transmissions, but it won't have much impact as node will have already received this message and see it as a stale message.

Indicate any segments of code that you have borrowed from the Web or other books.

The code to calculate Dijkstra's least cost path (def dijkstra(nodesDict: dict, current: str)) has been borrowed from -

https://gist.github.com/amitabhadey/37af83a84d8c372a9f02372e6d5f6732